THE COMBINATIONAL COMPLEXITY OF EQUIVALENCE

C. P. SCHNORR

Fachbereich Mathematik, Universität Frankfurt, Frankfurt, German Federal Republic

Communicated by M. Paterson
Received January 1975
Revised April 1975

Abstract. Consider the Boolean functions $\text{and}(n) = \bigwedge_{i=1}^{n} x_i$, $\text{nor}(n) = \bigwedge_{i=1}^{n} \neg x_i$ and the equivalence $\text{Eq}(n) = \text{and}(n) \vee \text{nor}(n)$. Let $L(F)$ be the smallest number of arbitrary binary Boolean operations that are used in any Boolean computation for F. We prove $L(\text{Eq}(n)) = 2n-3$, $L(\text{and}(n), \text{nor}(n)) = 2n-2$. There exist many structurally different optimal computations for $\text{Eq}(n)$.

1. Introduction

Let $K = \{0, 1\}$ be the set of Boolean values "0 ~ false" and "1 ~ true". Let $V = \{x_i \mid i \in \mathbb{N}\}$ be a countable set of Boolean variables. Let Ω be the set of all Boolean functions with variables in V. We consider Boolean computations that are based on the set of all 16 binary Boolean operations $w : \Omega^2 \to \Omega$ that correspond to binary Boolean functions $\tilde{w} : K^2 \to K$.

A Boolean computation β is a finite, directed, acyclic graph with labelled nodes such that

(1) every node v of β has either 2 or 0 entering edges.
The nodes without entering edges are called entries of β.

(2) every entry v of β is labelled with a variable $\text{op}(v) \in V$.

(3) every non-entry v of β is labelled with a binary Boolean operation $\text{op}(v)$. The entering edges of v correspond in a fixed ordered way to the arguments of $\text{op}(v)$.

With every node v of a Boolean computation we associate the output-function $\text{res}_\beta^v \in \Omega$. res_β^v is recursively obtained by applying $\text{op}(v)$ to the results of the two nodes that precede v. Initially $\text{res}_\beta^v = x_i$ for every entry v that is labelled with $x_i \in V$. We say β computes the functions $\text{res}_\beta^v \in \Omega$ for $v \in \beta$.

With $F \subset \Omega$ we associate the minimal number $L(F)$ of binary Boolean operations in any Boolean computation for F. $L(F)$ is called the combinational complexity (network complexity) of F.

Our main results are: $L(\mathrm{Eq}(n)) = 2n-3$, $L(\mathrm{and}(n), \mathrm{nor}(n)) = 2n-2 = L(\mathrm{and}(n))+L(\mathrm{nor}(n))$. This is the first non-trivial example of an exact evaluation of the network complexity in a case that there exist many structurally different optimal computations. For instance,

$$\bigwedge_{i=1}^{n-1}[x_i \ominus x_{i+1}] \quad \text{and} \quad \bigwedge_{i=2}^{n}[x_1 \ominus x_i]$$

yield optimal computations for $\mathrm{Eq}(n)$. The functions $\mathrm{and}(n)$, $\mathrm{nor}(n)$ yield a basic example of two independent functions (in the sense that $L(f,g) = L(f)+L(g)$) which depend on the same set of variables. The technique of this paper can be used to evaluate the network complexity of all Boolean functions with at most two prime implicants.

It has been proved in [2] that without using the binary operations $\ominus$ and $\oplus$ we need at least $2n-1$ binary operations for $\mathrm{Eq}(n)$. Hence the operation $\ominus$ helps in computing $\mathrm{Eq}(n)$ it does not help in computing $\{\mathrm{and}(n), \mathrm{nor}(n)\}$.

2. The combinational complexity of equivalence

Let us first summarize some of our supplementary concepts.

The partial ordering $f \leqslant g$ for $f, g \in \Omega$ is defined as:

$$f \leqslant g \Leftrightarrow f \wedge g = f.$$

Throughout the paper we shall only consider those computations β where there is just one entry for each input variable. So we shall identify entries and input variables of β. Moreover, we shall always restrict our considerations to those binary Boolean operations that depend on both of its arguments. Obviously both restrictions do not influency the combinational complexity $L(f)$.

A node μ is called a *successor* of node v if there is an edge from v to μ in β. Suppose we fix some output res_β^v of β to $i \in K$. This makes the node v computationally useless and we can eliminate v and all its successors μ by an appropriate change of the operations $\mathrm{op}(\alpha)$ of all successors α of μ. This yields a new computation $\bar{\beta}$. We say $\bar{\beta}$ is "obtained from β by fixing res_β^v to i".

Observe that every computation β is a directed graph. So the notion of a (directed) path in β should be clear. Let w be a path from v to μ then the node μ is called the head and v is called the tail of w. We abbreviate $f \wedge g$ as fg and $\neg f$ as $\bar{f}$.

With a Boolean computation β we associate the number $c(\beta)$ of nodes in β that are non-entries. $c(\beta)$ is the number of binary operations in β. Let $\deg_\beta(x_i)$ be the total number of edges that leave the entry v with $\mathrm{op}(v) = x_i$.

If $\deg_\beta(x_i) \geqslant 2$ for $i = 1, \ldots, n$ then $c(\beta) \geqslant 2n-1$ since $2n-1$ binary operations are necessary in order to collect the $2n$ edges at the inputs of β. Therefore we consider those input variables x_i with $\deg_\beta(x_i) = 1$.

Lemma 2.1. *Let β be an optimal computation for* $\mathrm{Eq}(n)$ *with* $n \geqslant 3$ *and* $\deg_\beta(x_n) = 1$ *and suppose the unique edge that leaves* x_n *enters an* $\ominus$*-gate* v*. Construct* β_i *from* β *by fixing the output* res^v_β *of* v *to the constant* $i \in K$ *for* $i = 0, 1$*. This implies*

(1) $c(\beta_i) \leqslant c(\beta) - 2$, $i = 0, 1$;

(2) *either* β_1 *computes* $\mathrm{Eq}(n-1)$ *or* β_0 *computes* $\mathrm{Eq}(n-1)$.

Proof. (1) Fixing $\mathrm{res}^v_\beta = i \in K$ eliminates the node v and all its successors.

(2) Let $h(x_1, x_2, \ldots, x_{n-1})$ be the input function of the other entry of node v. Then $\mathrm{res}^v_\beta = x_n \ominus h(x_1, x_2, \ldots, x_{n-1})$ and β yields a representation of $\mathrm{Eq}(n)$ as:

(I) $\quad \mathrm{Eq}(n) = f(x_n \ominus h(x_1, \ldots, x_{n-1}),\ x_1, x_2, \ldots, x_{n-1})$

with $f \in \Omega$. Let y be the first variable of f then we consider the unique decomposition of f:

(II) $\quad f = yf_1(x_1, \ldots, x_{n-1}) \vee \bar{y} f_0(x_1, \ldots, x_{n-1})$ with $f_1, f_0 \in \Omega$.

We define $d_1, d_0 \in \Omega$ as follows:

(III) $\quad d_1(x_1, \ldots, x_n) := (x_n \ominus h(x_1, \ldots, x_{n-1}))f_1(x_1, \ldots, x_{n-1})$,

(IV) $\quad d_0(x_1, \ldots, x_n) := (x_n \oplus h(x_1, \ldots, x_{n-1}))f_0, \ldots, x_{n-1})$.

It follows immediately from (I)–(IV) that

(V) $\quad \mathrm{Eq}(n) = \bigwedge_{i=1}^{n} x_i \vee \bigwedge_{i=1}^{n} \neg x_i = d_0(x_1, \ldots, x_n) \vee d_1(x_1, \ldots, x_n)$.

Observe that $x_n \oplus h = \neg(x_n \ominus h)$.

It follows from (V) that one of the following cases (1)–(4) must occur.

$$d_1(x_1, \ldots, x_n) = \mathrm{Eq}(n). \tag{1}$$

We know from (III) that $f_1(1, 1, \ldots, 1) = f_1(0, 0, \ldots, 0) = 1$ in this case. Hence there exists $u \in \Omega$ such that

$$f_1(x_1, \ldots, x_{n-1}) = \mathrm{Eq}(n-1) \vee u, \qquad u \wedge \mathrm{Eq}(n-1) = 0.$$

On the other hand it follows from (III) and $d_1 = \mathrm{Eq}(n)$:

$$h \wedge u \leqslant \mathrm{Eq}(n-1), \qquad \bar{h} \wedge u \leqslant \mathrm{Eq}(n-1).$$

Hence $u = 0$. This proves $f_1 = \mathrm{Eq}(n-1)$. Therefore β_1 computes $\mathrm{Eq}(n-1)$ in this case. Observe that β_1 is obtained from β by fixing $x_n \ominus h$ to 1.

$$d_0(x_1, \ldots, x_n) = \mathrm{Eq}(n). \tag{2}$$

This case is symmetric to case (1). It follows that $f_0 = \mathrm{Eq}(n-1)$ in this case. This proves that β_0 computes $\mathrm{Eq}(n-1)$ in this case.

$$d_0(x_1, \ldots, x_n) = \bigwedge_{i=1}^{n} x_i, \qquad d_1(x_1, \ldots, x_n) = \bigwedge_{i=1}^{n} \neg x_i. \tag{3}$$

It follows from (IV) that

$$\text{(IV')} \quad \begin{cases} x_n \bar{h}(x_1, \ldots, x_{n-1}) f_0(x_1, \ldots, x_{n-1}) = \bigwedge_{i=1}^{n} x_i, \\ \bar{x}_n h(x_1, \ldots, x_{n-1}) f_0(x_1, \ldots, x_{n-1}) = 0. \end{cases}$$

Hence $\exists u \in \Omega$:

$$f_0 = \bigwedge_{i=1}^{n-1} x_i \vee u \quad \text{with } u \leqslant \neg \bigwedge_{i=1}^{n-1} x_i.$$

Then (IV') implies $\bar{h}u \leqslant \bigwedge_{i=1}^{n-1} x_i$ and $hu \leqslant 0$. Hence $u = 0$. This proves $f_0 = \bigwedge_{i=1}^{n-1} x_i$ and by symmetry we have $f_1 = \bigwedge_{i=1}^{n-1} \neg x_i$. Therefore h can be substituted by 0 in this case. This proves that β is not an optimal computation for Eq (n).

$$d_0(x_1, ..., x_n) = \bigwedge_{i=1}^{n} \neg x_i, \quad d_1(x_1, ..., x_n) = \bigwedge_{i=1}^{n} x_i. \tag{4}$$

This case is symmetric to case (3) and it follows that β is not an optimal computation for Eq(n). □

Let x_i, x_j be two variables of a computation β. A pair of paths (w, v) is called an (x_i, x_j)-path if w and v have the same head and w starts at the variable x_i and v starts at the variable x_j and if w, v are edge-disjoint (see Fig.1).

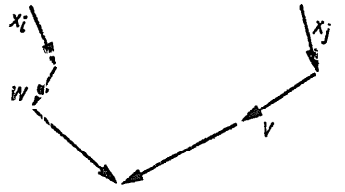

Fig. 1. Figure of an (x_i, x_j)-path (w, v).

Let $\Gamma(w, v)$ be the sum of the length of w and v and let $\Gamma_\beta(x_i, x_j)$ be the minimum on $\Gamma(w, v)$ for all (x_i, x_j)-paths (w, v).

Lemma 2.2. *Let β be any computation for* Eq(n). *Suppose that x_i, x_j are not inputs of any $\ominus$-gates or any $\oplus$-gates in β and that there exists a unique (x_i, x_j)-path in β. Then we can transform β into a new computation $\bar{\beta}$ for* Eq (n) *with $c(\bar{\beta}) = c(\beta)$ and $\Gamma_{\bar{\beta}}(x_i, x_j) < \Gamma_\beta(x_i, x_j)$ and $\bar{\beta}$ also contains a unique (x_i, x_j)-path.*

Proof. Suppose (w, v) is the unique (x_i, x_j)-path in β. Obviously $\Gamma(w, v) \geqslant 2$ and we get a contradiction in the case $\Gamma(w, v) = 2$. In this case the computation β yields a representation of Eq(n) as Eq(n) $= h(x_i \circ x_j, x_v, v \neq i, j)$ with $h \in \Omega$ where $\circ$ is the operation at the common head of w and v. This is clearly impossible. Hence this case cannot occur.

Now suppose $\Gamma(w, v) > 2$. In this case we reduce $\Gamma(w, v)$. Let μ be the successor of the x_i-node in w (if μ is the head of w, then consider the successor of the x_j-node in v). The assumption of Lemma 2.2 guarantees:

(1) the node μ has at most one leaving edge,

(2) the other input function of node μ does not depend on x_j, x_i.

Let $h(x_v \mid v \neq i,j)$ be the second input function of node v.
The computation β yields a representation of Eq(n) as

$$\mathrm{Eq}(n) = f(x_i \circ h,\ x_v, v \neq i) \quad \text{with } f \in \Omega,$$

where $\circ$ is op(μ).

Now suppose $\circ$ is $\wedge$. The other cases where $\circ$ is neither $\ominus$ nor $\oplus$ can be handled similarly.

Fact 1.

$$\bigwedge_{v \neq i,j} x_v \vee \bigwedge_{v \neq i,j} \neg x_v = 1 \Rightarrow h(x_v \mid v \neq i,j) = 1.$$

Proof. Suppose $h(x_v \mid v \neq i,j) = 0$. This implies

$$x_i \wedge h(x_v \mid v \neq i,j) = 0,$$

and therefore eliminates the dependence of β on x_i. This yields a contradiction since Eq(n) depends on x_i even under the restriction

$$\bigwedge_{v \neq i,j} x_v \vee \bigwedge_{v \neq i,j} \neg x_v = 1.$$

We now transform β into a new computation $\bar{\beta}$ for Eq(n) as follows. Suppose $\mathrm{res}^r_\beta = \mathrm{Eq}(n)$.

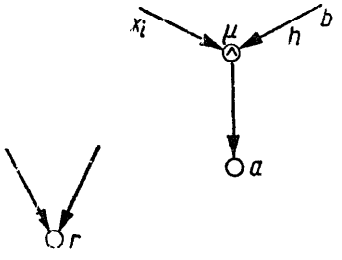

Fig. 2. Figure of β.

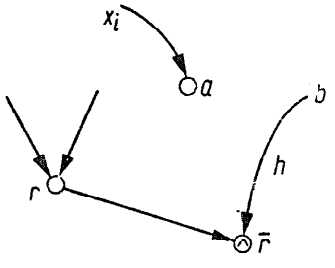

Fig. 3. Figure of $\bar{\beta}$.

Construction of $\bar{\beta}$.

(1) Substitute the output function res^μ_β by x_i, this eliminates the node μ.

(2) Adjoin a new $\wedge$-gate $\bar{r}$ to β with the input functions res^r_β and h.

Fact 2. $\mathrm{res}^r_\beta = \mathrm{res}^{\bar{r}}_{\bar{\beta}} = \mathrm{Eq}(n)$.

Proof. In the case $h(x_v \mid v \neq i,j) = 0$ we know from Fact 1 that Eq(n) takes the value 0. Hence $\mathrm{res}^r_\beta(x_1, \ldots, x_n) = \mathrm{res}^{\bar{r}}_{\bar{\beta}}(x_1, \ldots, x_n) = 0$ in this case. In the case $h(x_v \mid v \neq \quad) = 1$ we know that $\mathrm{res}^\mu_\beta = x_i$. Hence $\mathrm{res}^r_\beta(x_1, \ldots, x_n) = \mathrm{res}^{\bar{r}}_{\bar{\beta}}(x_1, \ldots, x_n)$. Therefore $\mathrm{res}^r_\beta(x_1, \ldots, x_n) = \mathrm{res}^{\bar{r}}_{\bar{\beta}}(x_1, \ldots, x_n)$.

Observe that our construction reduces the value $\Gamma_\beta(x_i, x_j)$ and that $\bar{\beta}$ also contains a unique (x_i, x_j)-path. □

Lemma 2.3. *Let β be a computation that depends on a set $V(\beta)$ of variables. Suppose that $\forall\, x_i, x_j \in V(\beta)$ there exist at least two different (x_i, x_j)-paths in β. This implies* $c(\beta) \geq 2 \cdot \|V(\beta)\| - 2$.

Proof. We proceed by induction on $\|V(\beta)\|$. The assertion is obvious for $\|V(\beta)\| = 2$. Now let $\|V(\beta)\| > 2$. The assumption of Lemma 2.3 implies that there exists an $x_v \in V(\beta)$ with $\deg_\beta(x_v) \geq 2$. Observe that "$\forall x_i \in V(\beta)$, $\deg_\beta(x_i) = 1$" would imply that for some variables x_i and x_j which are inputs of the same gate b there exists a unique (x_i, x_j)-path. This contradicts to the assumptions of Lemma 2.3.

Construct $\bar{\beta}$ from β by fixing x_v. This implies $\|V(\bar{\beta})\| = \|V(\beta)\| - 1$. Since $\deg_\beta(x_v) \geq 2$ it follows that $c(\beta) \geq c(\bar{\beta}) + 2$. The induction hypothesis implies $c(\bar{\beta}) \geq 2\|V(\bar{\beta})\| - 4$. Hence $c(\beta) \geq 2 \cdot \|V(\beta)\| - 2$. □

Main Theorem 2.4. $L(\mathrm{Eq}(n)) \geq 2n-3$.

Proof. We proceed by induction on n. Obviously $L(\mathrm{Eq}(2)) = 1$. Now suppose $n \geq 3$ and let β be an optimal computation for $\mathrm{Eq}(n)$. We consider 3 cases:

(a) "There exists a variable x_i with $\deg_\beta(x_i) = 1$ that enters either a $\ominus$-gate or a $\oplus$-gate."

It follows from Lemma 2.1 that in this case there exists a computation $\bar{\beta}$ for $\mathrm{Eq}(n-1)$ with $c(\bar{\beta}) \leq c(\beta) - 2$. The induction hypothesis implies $c(\bar{\beta}) \geq 2n-5$. Hence $c(\beta) \geq 2n-3$.

(b) "For all $x_i, x_j \in V(f)$ with $x_i \neq x_j$ there exist at least two (x_i, x_j)-paths."

In this case Lemma 2.3 implies $c(\beta) \geq 2 \cdot \|V(f)\| - 2$ for $f = \mathrm{Eq}(n)$.

Observe that whenever neither (a) nor (b) holds then we have the following case:

(c) "There exist nodes x_i, x_j such that there exists a unique (x_i, x_j)-path in β and x_i, x_j are not inputs of any $\ominus$-gate and any $\oplus$-gate in β."

In this case we apply Lemma 2.2 and we decrease $\Gamma_\beta(x_i, x_j)$.

If neither case (a) nor case (b) holds for this new computation, then we can still decrease $\Gamma_\beta(x_i, x_j)$ by applying Lemma 2.2. Observe that the property of having a unique (x_i, x_j)-path is preserved in the construction of Lemma 2.2.

Since we cannot decrease $\Gamma_\beta(x_i, x_j)$ below 3, a successive application of Lemma 2.2 finally constructs a computation β for $\mathrm{Eq}(n)$ with $c(\beta) = c(\beta)$ such that either case (a) or case (b) holds. □

Theorem 2.5. *The functions* $\mathrm{and}(n)$, $\mathrm{nor}(n)$ *are independent in the sense:* $L(\mathrm{and}(n), \mathrm{nor}(n)) = L(\mathrm{and}(n)) + L(\mathrm{nor}(n)) = 2n-2$.

The proof of this theorem uses the same technique as the proof of Theorem 2.4. So we leave the details to the reader. Lemmas 2.1 and 2.2 hold if $\mathrm{Eq}(n)$ is replaced by the set $\{\mathrm{and}(n), \mathrm{nor}(n)\}$. The discussion of the various cases is only slightly different. Using these corresponding versions of Lemmas 2.1, 2.2, the proof of Theorem 2.4

also applies to Theorem 2.5. Observe that $L(\text{and}(2), \text{nor}(2)) = 2$ guarantees the initial step of the induction on $L(\text{and}(n), \text{nor}(n)) = 2n-2$.

Finally we note that the technique of this paper can be applied to determine the combinational complexity of all Boolean functions with at most 2 prime implicants, i.e., of all functions $t \vee s \in \Omega$, where t,s are products of variables and negated variables.

Conclusion. The functions and(n), nor(n) are independent in a strong sense. Although our proof cannot directly be applied to Boolean computations with k-ary Boolean operations for $k > 2$, we conjecture that the independence of and(n), nor(n) still holds. This independence is based on the specific geometrical structure of these functions. We think that it is important to prove more general properties that make sets F of Boolean functions independent in the sense that $L(F)$ "approximately" equals $\sum_{f \in F} L(f)$ $(L(\bigvee_{f \in F} f)$, resp.).

References

[1] W. J. Paul, 2.25N-lower bound on the combinational complexity of Boolean functions, Preprint, Cornell University, Ithaca, N.Y. (1974).

[2] C. P. Schnorr, Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen, Computing 13 (1974) 155–171.